University of Central Punjab

Faculty of Information Technology

Data Structures and Algorithms
Fall 2021

Lab 08	
Topic	• Doubly LinkedList • Doubly LinkedList Applications
Objective	• The basic purpose of this lab is to implement Doubly LinkedList with ADT and test its applications.

Instructions:

- Indent your code.
- Comment your code.
- Use meaningful variable names.
- Plan your code carefully on a piece of paper before you implement it.
- Name of the program should be same as the task name. i.e. the first program should be Task_1.cpp
- **void main() is not allowed. Use int main()**
- **You have to work in multiple files. i.e separate .h and .cpp files**
- **You are not allowed to use system("pause")**
- **You are not allowed to use any built-in functions**
- **You are required to follow the naming conventions as follow:**
  - **Variables:** firstName; (no underscores allowed)
  - **Function:** getName(); (no underscores allowed)
  - **ClassName:** BankAccount (no underscores allowed)

Students are required to complete the following tasks in lab timings.

Task 1

Doubly Linked List is a type of linked list in which each node apart from storing its data has two links. The first link points to the previous node in the list and the second link points to the next node in the list. The first node of the list has its previous link pointing to NULL similarly the last node of the list has its next link pointing to NULL.

Your task is to create a generic abstract class named **DoublyLinkedList** with the following:

Attributes:

- ✓ Node <Type> *head;

Functions:

virtual void insertAtFront(Type) = 0;

Adds the element of type Type at the head of the doubly linkedlist.

virtual void insertAtEnd(Type) = 0;

Adds the element of type Type at the tail of the doubly linkedlist.

virtual Type removeFromFront() =0;

Removes and returns the first element of the doubly linked list and reduces size of the doubly linked list by 1.

virtual Type removeFromEnd() =0;

Removes and returns the element at the tail of the doubly linked lists and reduces size of the doubly linked list by 1.

Task 2

Using the abstract class, make your own **MyDoublyLinkedList** class having following functionalities:

- ✓ void insertAtFront(Type)
- ✓ void insertAtEnd(Type)
- ✓ Type removeFromFront()
- ✓ Type removeFromEnd()
- ✓ bool insertAt(int index, Type) -> insert's value at the index's position in the linked list.
- ✓ bool Search(Type) -> returns true if value is found in the linked list and false if the value isn't present.
- ✓ void Display() -> should display the linked list.

Your program should be menu based and should ask the user what action he wishes to perform.

Task 3

Implement **Stack** (LIFO) using **Doubly Linked Lists** only.

It should be a menu driven program which should first ask what functions of stack you would like to call. Then, operations related to stack may be called.

Functionality of Stack:

1. Push() // adds on top
2. Pop() // removes from top
3. Peek() // returns top most element
4. display() // displays contents of stack

Task 4

Using the class **MyDoublyLinkedList**, create the following additional functions:

- **reverseList()** // reverse the contents of the list.
- **getTotalSum()** // tells the sum of the elements in the list.
- **swapHeadAndTail()** // swaps the value present at the head and tail.
- **display()** // displays the list.
- **removeDuplicates()** // deletes any duplicate nodes present in the list. Ideally, the list should only be traversed once.
- **shuffleMerge()** // given two lists, merge their nodes together to make one list, taking nodes alternately between the two lists. So shuffleMerge() with {1, 2, 3} and {7, 13, 1} should yield {1, 7, 2, 13, 3, 1}. If either list runs out, all the nodes should be taken from the other list.